CAP 931 – Sales Agent Prototype Documentation

Student: Heba Abdelhadi
Project: AI Sales Assistant – Account Intelligence
Duration: 2 Days

1. Technical Setup

This project implements a prototype AI Sales Assistant that generates account intelligence for sales representatives.

Tech Stack

- **Python**
- **Streamlit (UI)**
- **FLAN-T5 Large (LLM)**
- **Requests & BeautifulSoup (web data extraction)**
- **PyPDF (PDF parsing)**

Files

- **app.py: Streamlit UI and input handling**
- **agents.py: Sales agent logic and prompt orchestration**
- **utils.py: Web and PDF text extraction**
- **llm.py: FLAN-T5 summarization logic**
- **config.py: API keys (excluded from GitHub using .gitignore)**

The application collects inputs via Streamlit, extracts public data from URLs and PDFs, summarizes it using an LLM, and outputs a one-page account intelligence report.

2. Time Management

Task	Time
Environment & Streamlit setup	2 hours
UI development	3 hours
Web & PDF extraction	4 hours
LLM prompt design	3 hours
Debugging & integration	3 hours
Documentation	1 hour
Total	16 hours

3. LLM Model Selection & Use

Model Used: FLAN-T5 Large

Justification:

- **Open-source and cost-efficient**
- **Strong instruction-following and summarization capabilities**
- **Suitable for structured business insights**
- **Avoids dependency on paid APIs**

Prompt constraints ensure the model only generates sales-related insights.

4. Inputs Handling

The Streamlit interface accepts:

- Product Name (required)
- Company URL (required)
- Product Category
- Value Proposition (required)
- Target Customer
- Competitor URLs
- Optional PDF upload

Validation ensures required fields are completed before report generation.

5. Data Integration & Output Relevance

- Public web data is extracted from company and competitor URLs
- PDFs are parsed to enrich product context
- Summaries are generated using LLM prompts
- Output is a single-page account intelligence report including:
  - Company Strategy
  - Competitor Mentions
  - Leadership Insights
  - Product Fit Summary
  - Sources

6. Challenges and Solutions

Challenge: Web scraping restrictions (403 errors)
Solution: Used corporate “About” pages and added browser headers.

Challenge: Inconsistent LLM responses
Solution: Added fallback summaries and minimum-length checks.

Challenge: API key security
Solution: Stored keys in config.py and added it to .gitignore.

7. Experiments

Different prompt formats were tested. Short, role-focused prompts produced the most relevant and concise one-page summaries.

8. System Outputs

The generated one-page account intelligence report is included via:

- Screenshots of the Streamlit output

Sources

- https://corporate.walmart.com/about

- https://corporate.target.com/about
- https://www.aboutamazon.com/about-us

9. Production Deployment Considerations

For production deployment:

- **Host Streamlit on cloud infrastructure**
- **Use environment variables for secrets**
- **Add caching, logging, and access controls**

Submission Checklist

Streamlit app runs
One-pager output generated
Screenshots included
Documentation file completed
API keys not committed
GitHub repo clean